
# Facing the Software Challenges of Multicore Designs

**Neil Puthuff**
Director of Hardware Engineering
Green Hills Software, Inc.
neil@ghs.com


Find Every Bug

# Overview

- There are significant problems in device software development that must be corrected
- These are not software-only problems; CPU hardware enforces a lack of visibility
- CPU hardware and debugging tools that provide an ideal solution will be presented

# Device Software Status Report

- Most software is of low quality
  - Shipped with bugs
  - Developed late
  - Missing features
- Problem is almost everywhere
  - Desktop PC's, cell phones, Mars, etc.
- Trends that make this situation worse:
  - Larger applications
  - Time-to-market pressure
  - **Multiple cores**

# Paths to improvement

- Coding standards, code inspection
  - Successfully used in avionics
- Enhanced tools and debug facilities
  - Individual core control
    - Reset / run control / register and memory access
  - Synchronized core control
    - Run, halt, step, cross-triggering on multiple cores
  - **Improved visibility**

**Debug facilities are too often an afterthought in processor designs**

# Visibility and Debugging

"I can't fix what I can't see"

- This includes most of what happens inside a CPU

- Solution: Real Time Trace
  - Provides complete visibility into software execution
  - Non-intrusive
  - Debug during full-speed execution

Too few processors have implemented real time trace, for a variety of reasons

# What is Real Time Trace?

**On-chip logic exporting data on core activities:**

- Instruction execution (instruction trace)
- Memory accesses (data trace)
- Software events, triggers, breakpoints

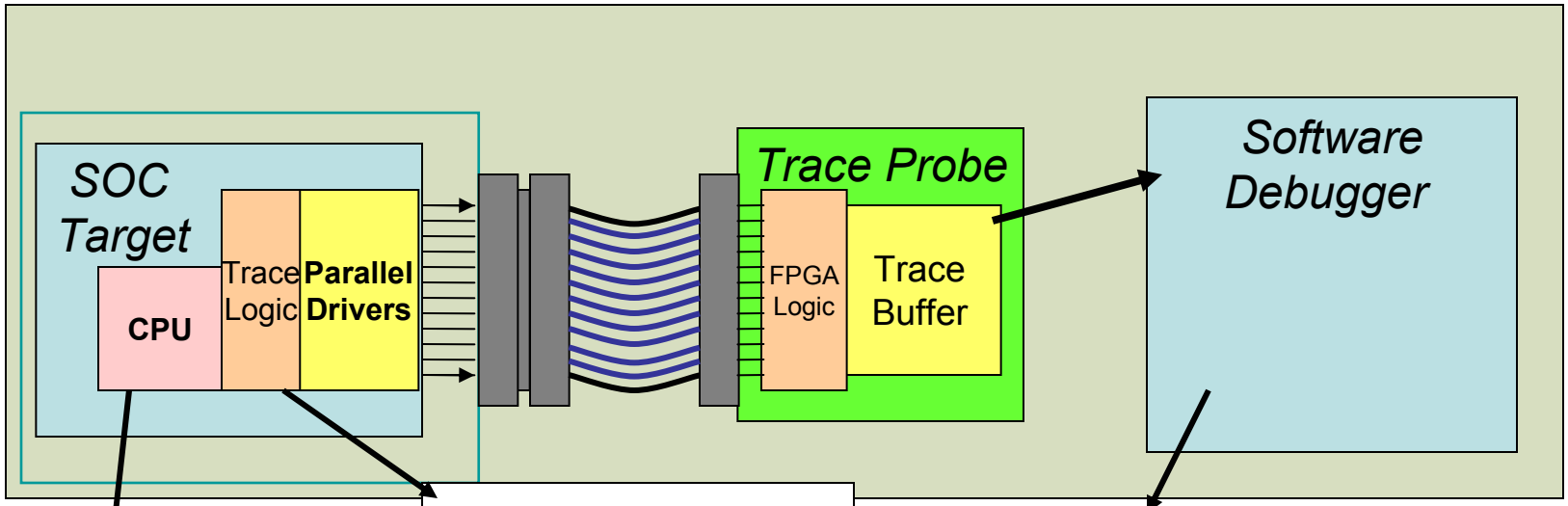
**Trace data is encoded to reduce bandwidth**

- Instruction trace: as low as 1 bit per clock cycle
- Data trace requires much greater bandwidth

**Trace data is exported to:**

- On-chip trace buffer
- External storage via dedicated trace port

# Example of Trace Data Flow




*Application*

```
ldr   r0,[r2]
add   r1,1
cmp   r1,0
bne   [0xffe0]
...
```

*Trace encoding and export*

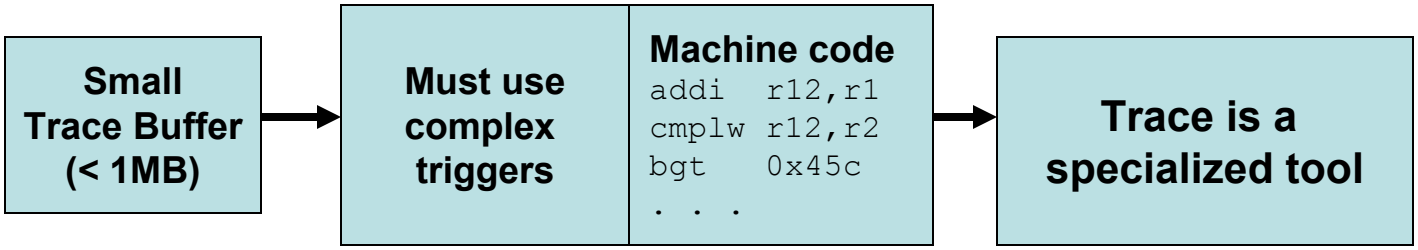
```
address=0x2014ac2c
next instruction
next instruction
next instruction
branch taken
...
```

*Correlation of trace info to application image*

```
0x2014ac2c   ldr   r0,[r2]
0x2014ac30   add   r1,1
0x2014ac34   cmp   r1,0
0x2014ac38   bne   [0xffe0]
...
```

# Historical Problems with Trace

## Small Trace Buffers




Solved with gigabyte-capacity trace probes

- Can store billions of instruction cycles

- Gigabyte capacities requires improved tools
  - Create virtual target from trace data
    - Like TiVo for debugging
  - Statistics, anomaly detection, etc

**Trace is finally suitable for everyday debugging**

# Misapplied Resources

- Traditional debugging:
  - Software engineer spends countless hours searching for bugs
  - Meanwhile, their desktop PC furiously burns CPU cycles doing nothing
- Trace debugging can turn this around:
  - Desktop PC should be tasked with *finding* bugs
  - Software engineer can spend more time *fixing* bugs

Think in terms of **massive** trace collection:
always enabled, every bug is captured

# Remaining Challenges for Trace

Other reasons why trace isn't widespread:

- High pin counts
- Difficult timing on fast targets
- High bandwidth needs
- Lack of independent unifying standards
- Fragmented support for proprietary trace ports
- Royalties, IP licensing fees
- Lack of system-level visibility
- Most trace ports don't scale well to multi-core

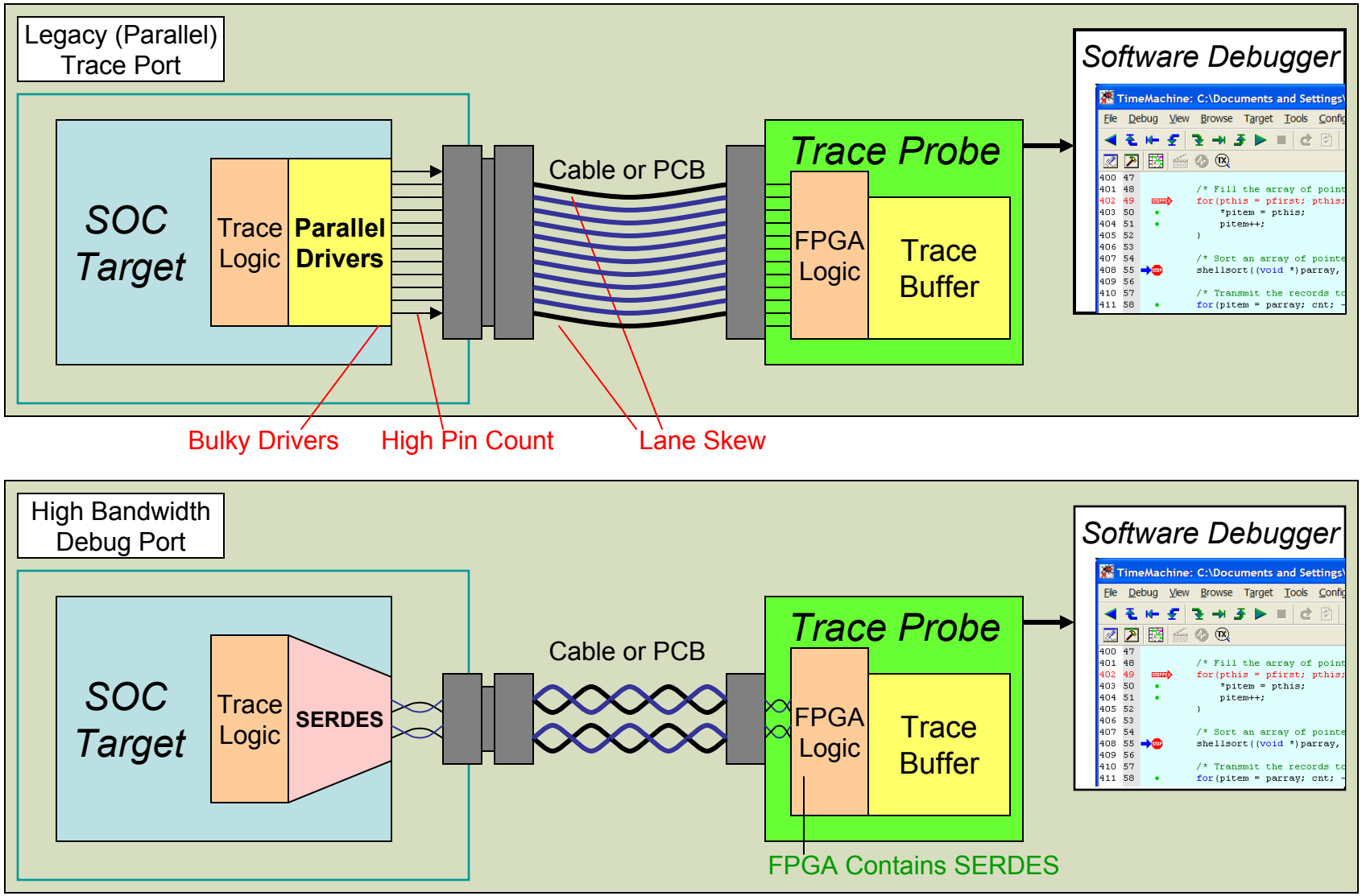
**There is a technology that meets these challenges…**

# Meeting the Challenges

## High Bandwidth Debug Port (HBDP)

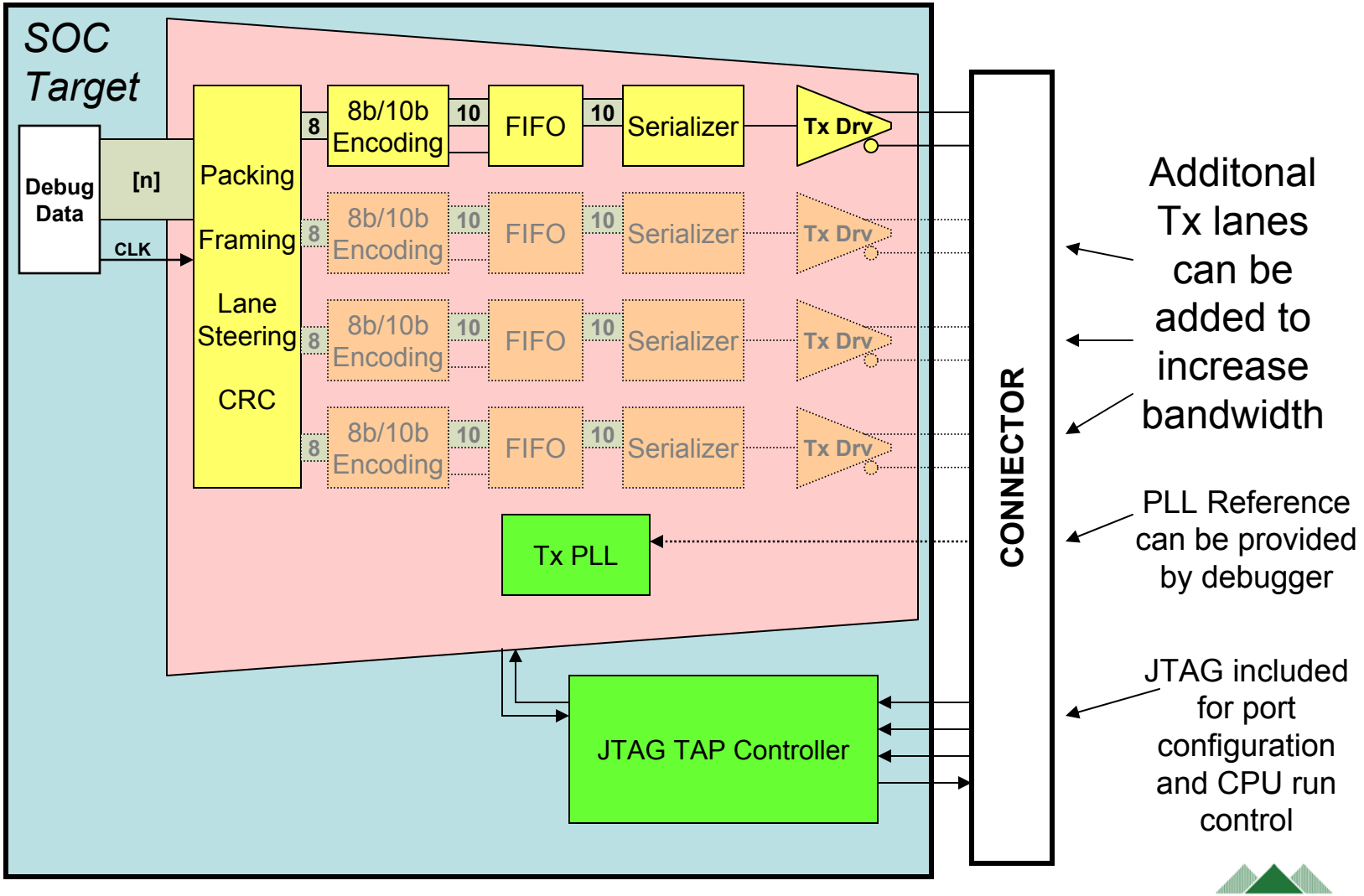
- Replaces parallel trace port with LVDS SERDES port
  - Provides high bandwidth with low pin count
  - Timing is lane-specific (clock embedded in data)
- Implementation-friendly
  - Supportable with COTS FPGA's (uses Xilinx Aurora protocol)
  - Synthesizable RTL available for silicon
  - Non-proprietary, on track to become IEEE/ISTO standard
- Supports multi-source debugging
  - Umbrella standard: Trace of multiple cores, system elements, hardware events
  - Applicable to trace ports from all leading architectures

# Concept Diagrams




Target Rx direction and JTAG run control omitted for clarity

# HBDP Tx Block Diagram



# Bandwidth vs. Lane Count and Speed

- Signaling rates from 622Mbps to 10.3125Gbps per lane supported by COTS FPGA's
- Flexible number of lanes

| | Effective Throughput in MBytes/Sec<br>Number of 300MHz cores traceable at 4b/1b per cycle | | | | |
|---|---|---|---|---|---|
| Link Data Rate | 1 Lane | 2 Lane | 3 Lanes | 4 Lanes | 8 Lanes |
| 622 Mbps | 61<br>0/1 | 123<br>0/3 | 184<br>1/4 | 246<br>1/6 | 491<br>3/13 |
| 2.5 Gbps | 246<br>1/6 | 494<br>3/13 | 741<br>4/19 | 988<br>6/26 | 1975<br>13/52 |
| 3.125 Gbps | 309<br>2/8 | 617<br>4/16 | 926<br>6/24 | 1234<br>8/32 | 2468<br>16/64 |
| 4.25 Gbps | 420<br>2/11 | 839<br>5/22 | 1259<br>8/33 | 1678<br>11/44 | 3358<br>22/89 |
| 10.3125 Gbps | 1238<br>8/32 | 2475<br>16/65 | 3712<br>24/98 | 4950<br>32/131 | 9900<br>65/263 |
| Notes: ~21% bandwidth lost to 8b/10b encoding, word sync, lane deskew, etc.<br>Highest data link rates to use 64/66b encoding | | | | | |

# HBDP Data Sources

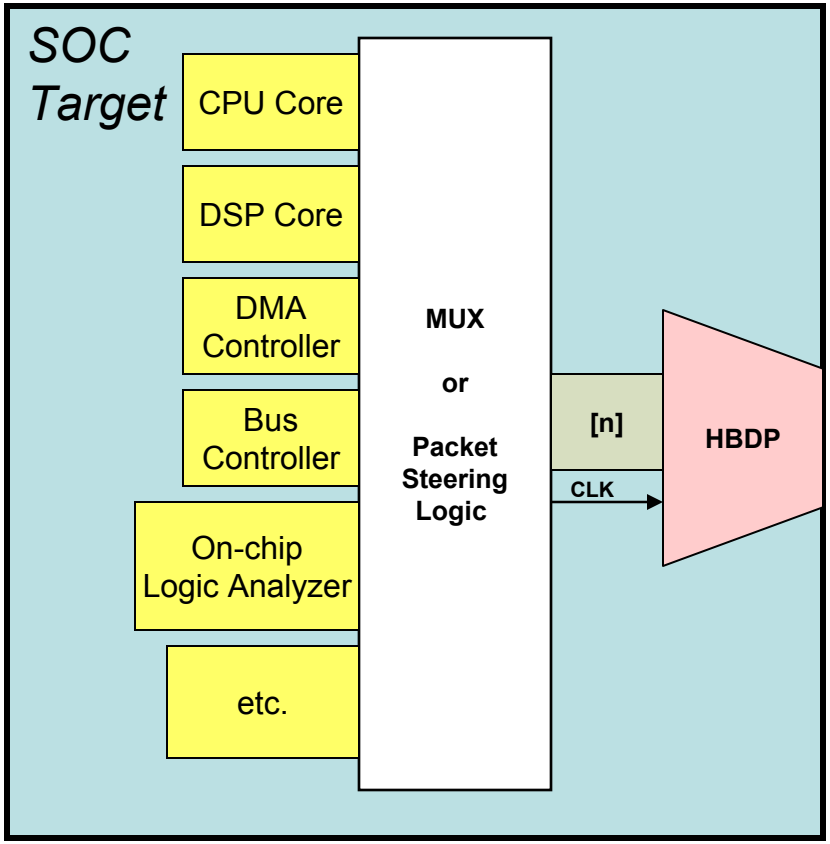
HBDP is a transport layer only

**Works with existing software debuggers**

Bandwidth usage flexibility to accommodate:

- CPU trace information
- Bus and DMA controllers
- On-chip logic analysis
- ???

**Suitable for a variety of development uses**

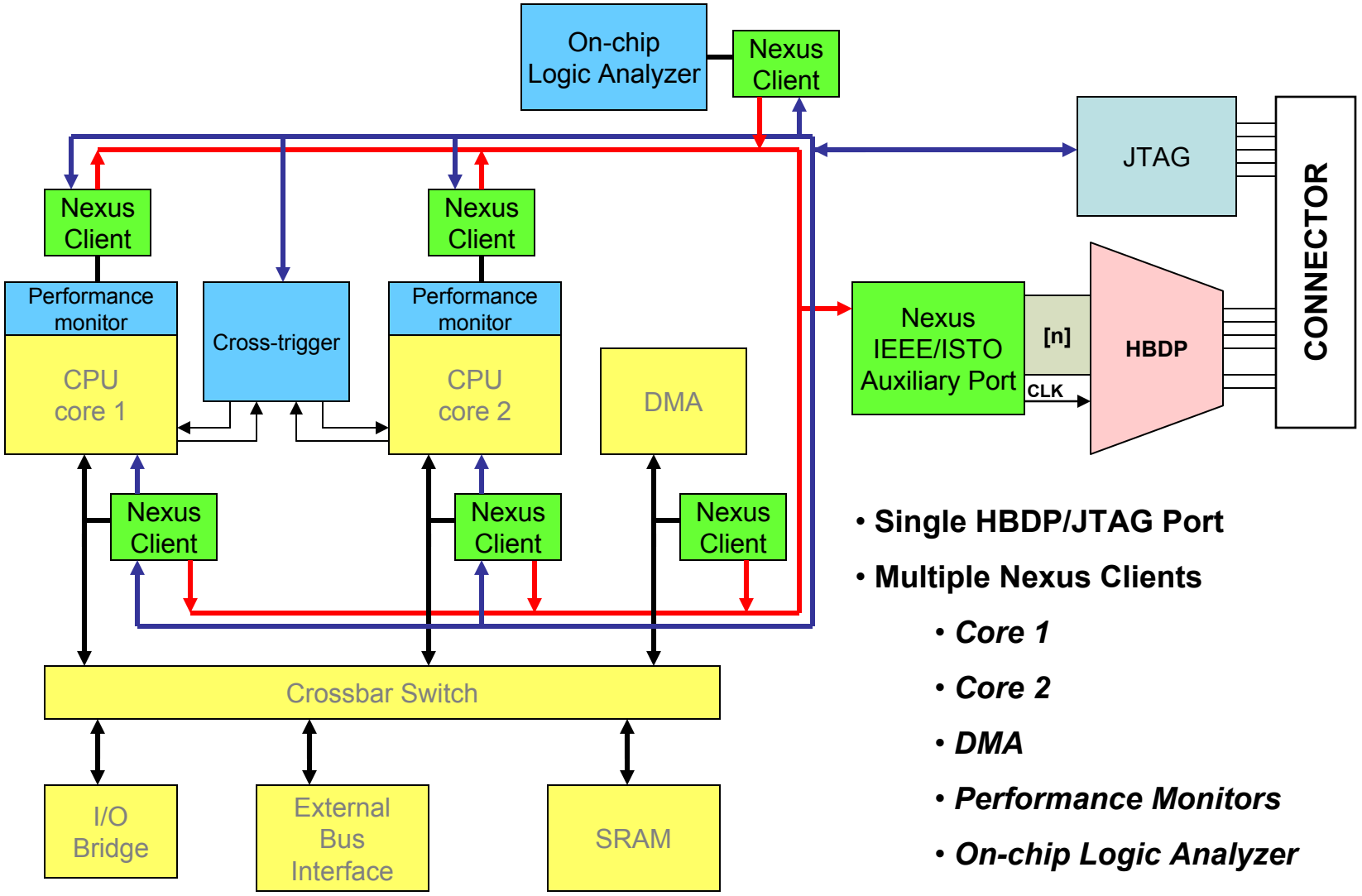


# Example: Nexus 5001 Interface


IEEE/ISTO-5001 Standard debug port

- Ready for Multicore:
  - Allows multiple Nexus clients to share debug/trace port
- Already deployed on PowerPC, ARM, DSP, and specialized processors
- Widely supported by debug tools vendors
- Complete specifications are freely downloadable:
  http://www.nexus5001.org

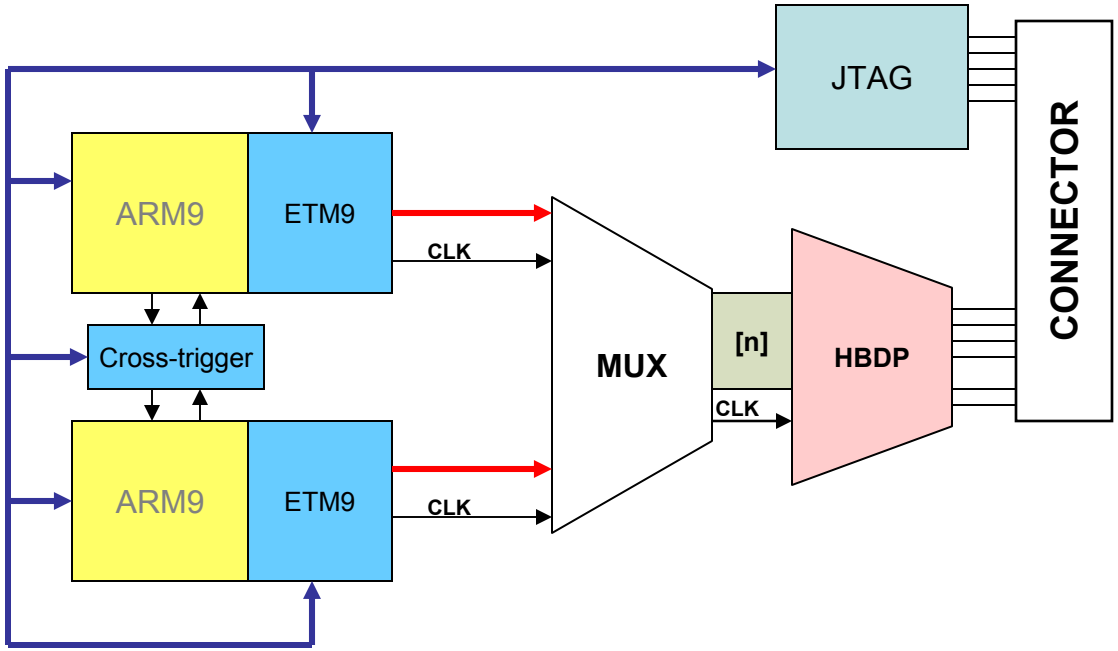
# HBDP with Nexus




- **Single HBDP/JTAG Port**
- **Multiple Nexus Clients**
  - ***Core 1***
  - ***Core 2***
  - ***DMA***
  - ***Performance Monitors***
  - ***On-chip Logic Analyzer***

# HBDP with ARM ETM

- Single HBDP/JTAG Port
- Target cores share trace port via multiplexer
- Data received by trace probe is unchanged by HBDP

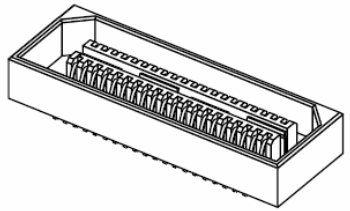



## Designed for Flexibility:

## Same connector, protocol, trace probe used across wide variety of targets and debug tasks

# HBDP Development Status

- Partnership development with several major CPU and SOC manufacturers
- Concept proven on several architectures
- Connector and cable testing
  - Lab / Desktop
  - Industrial / Automotive
  - Minimum pin count / board space



Samtec QSE / QTE series

- Draft Specification

# HBDP Summary

- Enables unprecedented system debug visibility
- Avoids classic pitfalls of real time trace
- Non-proprietary: to be open standard via IEEE/ISTO-5001
- Uses proven technology for simplified design-in
- Minimizes silicon overhead on device implementing the port
- Flexible: Works with existing trace ports and debuggers
- Scalable in link speed and lane count to fit needs
- Devices and trace probes now in development for concurrent release

**High Bandwidth Debug Port meets the challenges for single- or multi-core system debugging**

For more information: neil@ghs.com